Journal of Systems Architecture 60 (2014) 726–740

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Challenges in real-time virtualization and predictable cloud computing

Marisol García-Valls [a,*], Tommaso Cucinotta [b], Chenyang Lu [c]

[a] Distributed Real-Time Systems Laboratory, Department of Telematics Engineering, Universidad Carlos III de Madrid, Av. de la universidad 30, 28911 Leganés, Madrid, Spain
[b] Bell Laboratories, Alcatel-Lucent, Blanchardstown Business and Technology Park, Snugborough Road, Dublin, Ireland
[c] Cyber-Physical Systems Laboratory, Department of Computer Science and Engineering, Washington University in St. Louis, 1 Brookings Dr., St. Louis, MO 63130, USA

ARTICLE INFO

Article history:
Received 10 May 2013
Received in revised form 20 July 2014
Accepted 30 July 2014
Available online 9 August 2014

Keywords:
Cloud computing
Soft real-time systems
Virtualization
Resource management
Quality of service
SLA

ABSTRACT

Cloud computing and virtualization technology have revolutionized general-purpose computing applications in the past decade. The cloud paradigm offers advantages through reduction of operation costs, server consolidation, flexible system configuration and elastic resource provisioning. However, despite the success of cloud computing for general-purpose computing, existing cloud computing and virtualization technology face tremendous challenges in supporting emerging soft real-time applications such as online video streaming, cloud-based gaming, and telecommunication management. These applications demand real-time performance in open, shared and virtualized computing environments. This paper identifies the technical challenges in supporting real-time applications in the cloud, surveys recent advancement in real-time virtualization and cloud computing technology, and offers research directions to enable cloud-based real-time applications in the future.

1. Introduction

The widespread availability for the masses of high-speed Internet connections at affordable rates, by means of DSL and more recently optical technologies, paired with an unprecedented connectivity through cellular and wireless technologies, is enabling an inescapable shift towards distributed computing models. Applications relying merely on physical resources and data available in the local *personal computer* (PC) are slowly but consistently becoming part of the history, as the PC declines leaving the way to a new era of distributed computing. This is subsumed into the recently expanding paradigm of Cloud Computing [87], in which resources are rented in an on-demand and pay-per-use fashion from cloud providers. Just as a huge hardware machine, cloud computing data centres deliver an infrastructure, platform, and software applications as services that are available to consumers. Such services are referred to as IaaS (*infrastructure as a service*), PaaS (*platform as a service*), and SaaS (*software as a service*), respectively [88]. Cloud applications are capable of running and spreading out their computations and data on as many nodes as needed, and they can access huge amounts of data directly available within the premises of cloud data centers. Shortly, cloud computing is enabling the next generation of computing services, heavily geared towards massively distributed and on-line computing, as well as enabling a new model of on-demand *high performance computing* (HPC) accessible to anyone from anywhere, whenever needed.

As new application domains enter the cloud world progressively, real-time systems are also expected to move in this direction due to the tremendous possibilities and augmented utility that this paradigm could bring about. Examples are both on the hard and soft sides of real-time systems, such as military distributed control systems applied to remote surveillance, early response and warning systems, unmanned vehicles with augmented intelligence from the sensor cloud, or cloud gaming, among others.

Cloud computing, and particularly the use of public clouds, brings advantages on the technical, environmental and business sides, allowing multiple under-utilized systems to be consolidated within fewer physical servers hosting them. A cloud provider can manage physical resources in a very efficient way by scaling on the several hundreds and thousands of customers (a.k.a., *tenants*) with dynamically changing workload requirements, by re-optimizing the infrastructure in a completely automated (or semi-automated) fashion whenever needed, providing high levels of availability and reliability. One of the most important technologies that enabled this paradigm shift in computing is *virtualization*, and particularly *machine virtualization*.

Machine virtualization (also referred to as processor virtualization) allows a single physical machine to emulate the behavior of multiple machines, with the possibility to host multiple and

* Corresponding author.
E-mail addresses: mvalls@it.uc3m.es (M. García-Valls), tommaso.cucinotta@alcatel-lucent.com (T. Cucinotta), lu@cse.wustl.edu (C. Lu).

http://dx.doi.org/10.1016/j.sysarc.2014.07.004
1383-7621/

heterogeneous *operating systems* (called guest operating systems or guest OSs) on the same hardware. A *virtual machine monitor* (VMM), or *hypervisor*, is the software infrastructure running on (and having full control of) the physical host and which is capable of running such emulation.

Virtualization allows for server consolidation in data centers, where multiple operating systems that would leave their underlying hosts under-utilized can be moved to the same physical resources. This enables the achievement of a reduction of the number of required physical hosts, and their improved exploitation at higher saturation levels, thus saving costs and energy [86].

The multi-tenant nature of cloud computing has a great influence on the increasingly rich and challenging user requirements on cloud infrastructures. Users demand not only access to on-line storage, but also to real-time and interactive applications and services. This is also witnessed by visionary products already on the market, such as lightweight computers which are almost incapable of doing anything locally, unless they are connected to the "Cloud".

In a *high-performance cloud computing* (HPCC) environment, applications have much stronger temporal requirements; as such, the characteristic of performance, including resource guarantees and timely provisioning of results, becomes critical. It is actually an open research area to match such requirements with virtualized environments due to I/O overhead and jitter of the required duration of executed instructions. Moreover, activities or jobs are mostly allocated to a specific core, but they often have synchronization dependencies with respect to other activities.

Merging cloud computing with real-time is a complex problem that requires to also focus on (among others) the efficient access to the physical platform. Although real-time hypervisors typically may allow applications to access to the physical machine, in virtualized environments for cloud computing, it is clear that the hardware is typically not directly accessible by the user-level application software layers. With the current available technology, it could be possible to improve service response times from a cloud platform using high performance techniques. The main problems lie on the multi-tenancy of the cloud computing platforms that execute on heavy loaded servers the requests of several independent users. Currently, there are a number of commercial real-time hypervisors (some providing hierarchical scheduling) for safety critical systems of different origins such as WindRiver, Acontis Technology, SysGO, OpenSynergy, LynuxWorks, or Real Time Systems GmbH. However, it is not likely to see them among the mainstream cloud technology in the immediate future due to performance levels and compatibility problems at the low level execution layer. Mainstream and real-time ones were created with different objectives. For example, real-time hypervisors were not invented for maximizing throughput of user requests and providing statistical guarantees on service contracts, but to preserve temporal isolation and determinism.

Cloud computing technology is, in origin, not targeted at hard real-time applications which typically run in closed environments. This paper targets at the significant challenges in applying cloud computing technologies to *soft real-time applications*. Examples of soft real-time domains are, for instance, online video streaming (e.g. Netflix on Amazon EC2), cloud-based gaming, and telecommunication management. Such applications can benefit significantly from cloud computing due to their highly dynamic workloads that desire elastic allocation of resources. Cloud-based gaming is gaining momentum in the market. Concrete instances such as Netflix running in Amazon EC2, and both Xbox and Playstation are planning to offer cloud-based gaming. For example, Microsofts Xbox One game console allows computation of environmental elements to be offloaded to the cloud; Sony recently acquired Gaikai, a major open cloud gaming platform. As more latency-sensitive games and players move toward the cloud, it is becoming increasingly important to meet varying latency requirements in the cloud computing environment. User studies [3], for example, show that networked games require short response delay, even as low as 100 ms, e.g., for first-person shooter games [2]. Table 1 provides delay tolerance for on-line gaming applications.

Table 1
Delay tolerance in traditional gaming [4].

Game type	Delay threshold (ms)
First person shooter	100
Role playing game	500
Real-time strategy	1000

In the telecommunication industry, there is a major shift from hardware-based provisioning of network functions to a software-based provisioning paradigm where virtualized network functions [94] are deployed in private or hybrid clouds of network operators [89]. For example, IP Multimedia Subsystem (IMS) components are traditionally designed and calibrated to run on specific hardware platforms with precise real-time and reliability requirements, given a target maximum workload specification, such as maximum number of supported subscribers or call attempts per second. Matching the same requirements in a virtualized context where a multitude of virtual machines (VMs) share the same physical hardware for providing a plethora of services with highly heterogeneous performance requirements to independent customers/end-users brings many challenges, some of which can be tackled as summarized in this paper.

For soft real-time applications, bypassing system software and directly exposing the hardware to applications is neither needed nor it may be the most productive approach. However, the integration of real-time scheduling policies within virtualization platforms produces a direct benefit, and the system can deliver real-time performance to the application in a hierarchical manner. In this paper, we describe the problems arising from mixing the requirements of soft real-time workloads when deployed in the context of distributed and virtualized physical infrastructures, such as in cloud computing. We describe the service level agreement notion and the challenges to support real-time attributes in them. The paper provides a survey that focuses on soft real-time applications that demand certain degrees of service level agreements in terms of real-time performance, but does not require hard real-time performance guarantees. We describe some of the available approaches to integrate the real-time model in a virtualized application model. We provide some approaches to HPCC, and the challenges introduced by the network communication as it requires I/O access incurring in extra delays; some solutions for improving the efficiency of the network are presented.

This survey is complementary to a recent review on real-time virtualization for embedded systems [98]. While [98] focused on hard real-time embedded systems, we address predictable and real-time performance issues in not only embedded systems, but also cloud computing systems, including approaches to soft real-time performance and QoS issues. The paper is structured as follows. Section 2 offers an overview of the virtualization technology for cloud computing focusing on the real-time issues that appear therein. Also, the type of virtualization approaches and their performance levels are provided to give an idea of the suitability for real-time domains. Section 3 presents the challenges for merging real-time and cloud computing: we provide a mapping of terminology between the cloud computing and the real-time worlds; we present specific issues concerning the access to the platform resources faced by virtual machine monitors and hypervisors; we overview different approaches proposed in the literature for scheduling virtual machines; and we explain the network challenges for achieving real-time support and some HPC techniques to tackle

them. Section 4 describes how some example projects have approached the design and development of virtualization solutions for cloud computing. Finally, Section 5 presents possible future directions of research in this area.

2. Cloud computing and virtualization

The cloud computing paradigm provides a number of benefits through the presence of its enabler, i.e., the virtualization technologies. Firstly, this section provides a high level overview of the benefits of cloud computing. In practice, the specific virtualization type is always a trade-off among the proximity of control and access to the actual underlying hardware platform, the performance offered to the hosted software, and the flexibility of the development and deployment. Before describing the different types of virtualization that is provided within this section, it is necessary to explain the architecture of virtual machines that is, in fact, closely related to the architecture of a hardware platform.

2.1. Benefits of virtualization

Virtualization technology offers applications an abstract view through interfaces of the underlying hardware platform and resources. As described in [10,11], virtualization has several benefits for enabling cloud computing:

- *Functional execution isolation.* The hypervisor handles the protection among virtual machines (VMs) and, therefore, among the applications on different VMs. Users can be granted privileges within their virtual machine without compromising the isolation or host integrity.
- Virtualization enables the provisioning of highly specialized and *customized environments* that may contain specific purpose operating systems, libraries, and run-time execution environments. In fact, virtualization offers functional isolation therefore enabling multiple views over the same physical hardware.
- *Easier management.* Customized run-time environments can be started up, migrated, shut down, in a very flexible way, depending on the needs of who provides the underlying hardware.
- *Coexistence of legacy applications* with brand new ones. VMs help to preserve binary compatibility in the run-time environments for legacy applications.
- *Testing and debugging parallel applications* can leverage virtualized environments, as a full distributed system may be emulated within a single physical host.
- Hypervisors and their live-migration capabilities allow for *enhancing reliability* of hosted virtualized applications, making them independent of the reliability of the underlying hardware, in a seamless and transparent manner for applications.

Nevertheless, the multi-tenancy nature of cloud computing, along with its higher consolidation level, constitutes also one of the factors raising many challenges still to be properly tackled. The increased level of sharing of physical resources among multiple software components and applications to be hosted on behalf of different customers makes it more and more difficult to provide stable and predictable performance levels to each one of them. Indeed, virtual machines (VMs) can be executed concurrently in a virtualized platform simultaneously competing for physical resources that are scheduled by an underlying hypervisor. VMs and activities/tasks within VMs adjust to a hierarchical scheduling view where time can be partitioned among VMs; within VMs, the processor is further granted to tasks or threads according to the guest OS specific scheduling policy.

The real-time domain can benefit from the same advantages of general purpose applications in the cloud, despite the drawbacks of the above mentioned characteristic of server consolidation. An example can be the construction of a private cloud for a remote control and operation of an industrial automation plant. The plant operation can be monitored through sensors and the information can be stored and analyzed in a private cloud. Also, the operation logic of some machines/instruments can be outsourced to a private cloud easing maintenance and remote control of the plant floor. Due to the real-time constraints of some operations, the latencies of the virtual nodes as well as the communication jitter have to be controlled.

2.2. Virtual machine architectures

2.2.1. Hardware–software interfacing

Software systems continue to evolve despite being implemented on top of hardware architectures that are increasingly complex. This is possible since computer systems follow a hierarchical design with well-defined interfaces that establish cleanly-separated levels of abstraction. It is well known that well structured and defined interfaces facilitate independent subsystem development by engineering teams targeting both software and hardware systems. An instruction set architecture (ISA) is a clear example of this.

However, there are some limitations to the pure usage of interfacing. For example, a binary format of an application is tied to a specific ISA and operating systems libraries. This was found limiting specially in a ultra-connected world where there is a clear benefit in the possibility of easily moving pieces of code from one machine to another. Machine virtualization [1] overcomes this limitation by adapting the interface and visible resources of a system onto the interface and resources of an underlying, possibly different, real system. The goal of virtualization technology is not to simplify or hide the internal details of a physical system. The goal of virtualization is to use the interfacing abstraction of the real hardware resources or subsystems as a way to map the virtual resource to the actual one.

2.2.2. Development of virtual execution environments

There are different approaches to develop virtual execution environments that depend on the fidelity with which they implement the different interfaces provided within the architecture of a computer system, i.e., hardware or instruction set architecture, operating system interface, application binary interface, or application programming interface. For example, in the case of JVM or Java Virtual Machine, a different instruction set architecture is provided that contains a dynamic translator or interpreter of the code programmed in a high level language as Java.

Therefore, a VM is an execution environment that is a software implementation of a physical execution platform, machine, or computer that can run programs just as the physical machine would do. A VM provides an abstraction of a hardware platform, a given operating system, and even a set of libraries/applications to be run. The VM then executes extra software layers that may introduce temporal penalties.

The design of virtual environments can be approached from two different sides [5]: hardware partitioning or hypervisor technology. Hardware partitioning subdivides the physical machine into different partitions each of which can execute a different operating system. Such partitions are typically created with coarse units of allocation [5], such as whole processor or physical board. This type of virtualization allows for hardware consolidation, but does not have the full benefits of resource sharing and emulation offered by hypervisors.

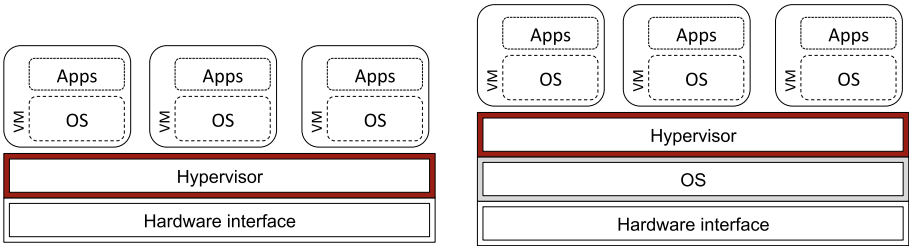

Fig. 1. Type 1 (bare metal) hypervisors versus Type 2 hypervisors.

2.2.3. Hypervisor types

The design and implementation of the hypervisor is of paramount importance because it has a direct influence on the throughput of VMs, which can be very close to that of the native hardware. Hypervisors are the primary technology of choice for system virtualization since it is possible to achieve high flexibility in how virtual resources are defined and managed. The replication provided by hypervisors is achieved by partitioning and/or virtualizing platform resources. As explained in [8], there are two main types of hypervisors (Fig. 1):

- *Bare metal or type 1* that runs directly on the physical hardware platform. It virtualizes the critical hardware devices offering several independent isolated partitions. It also provides basic services for inter-partition control and communication. Type 1 hypervisors are suitable for real-time systems since virtual environments using type 1 hypervisors are close to the hardware and are able to use hardware resources directly.[1]
- *Type 2* hypervisors that run on top of an operating system that acts as a host. These are hosted hypervisors since they run within a conventional operating system environment. The hypervisor layer is typically a differentiated software level on top of the host operating system (that runs directly above the hardware), and the guest operating system runs at a different level.[2]

The open-source Xen [9] hypervisor brought in the concept of *para-virtualization*, a technique allowing for replacing the trap-based emulation of privileged instructions and virtualized peripherals with direct calls to *hypercalls* provided by the hypervisor. Namely, the virtualized machine is enriched with a special API that can be used by modified guest operating systems to communicate efficiently with the hypervisor, similarly to how an application uses system calls to communicate with the kernel of an operating system. The code running inside partitions (normally, the guest operating system) must be aware of being running in a virtualized environment, and accordingly use the hypercalls whenever needed. Running on bare metal, the Xen hypervisor supports guest operating systems in VMs named domains, where domain 0 is used for physical I/O. Xen reduces the overhead of full system virtualization which consists of fully emulating the underlying hardware to run unmodified operating systems. Consequently, para-virtualization is more suitable for real-time cloud computing since it identifies the specific components of an operating system that have to be virtualized in order to optimize performance. However, this brings in other challenging points such as virtualizing and sharing memory between guest operating systems. Examples are guest operating systems that leverage characteristics built in hardware (e.g., TLB of x86) or handing privileged instruction calls made by guests operating systems and exceptions generated by the hardware.

There have been recent efforts on virtualization for real-time and embedded systems. For example, both XtratuM[3] [6,7] and VLX[4] are bare metal hypervisors designed to support embedded systems through para-virtualization, and RT-Xen[5] is an open-source real-time virtualization platform. XtratuM is based on Xen, and it supports partitioned systems following the principles of hierarchical scheduling; it complies with the ARINC standard and guarantees temporal and spatial isolation targeted at critical software systems. RT-Xen extends the Xen hypervisor by introducing real-time VM schedulers designed based on hierarchical and compositional real-time scheduling [46,47]. RT-Xen also features a novel communication architecture in the manager domain to support real-time communication among VMs sharing a physical host [55]. On the other hand, outside of the embedded world in the context of predictable cloud computing, the IRMOS European project investigated on how to enable predictability in cloud services by proper use of development tools, modeling techniques and real-time scheduling on the underlying physical resources [27]. More details on IRMOS will follow in Section 4.

2.3. Virtualization techniques and performance characteristics

The various techniques to virtualization environments are used in different works in slightly different ways and even with some semantic overlapping. In this section, the different types of virtualization classifications that can be found in the literature are put forward and summarized, and references to their different performance characteristics are given.

2.3.1. Full virtualization

Full virtualization allows for running unmodified guest operating systems by full emulation of the hardware that they believe to be running on top of, e.g., network adapters and other peripherals. This way, it is easily possible to run multiple operating systems, even heterogeneous ones, on the same hardware. Then, the hypervisor takes care of the necessary network emulation, in order to let the VMs communicate with the outside world and with each other. Full virtualization is often too expensive, causing continuous traps to the hypervisor that intercepts the special privileged instructions that a guest operating system kernel believes to execute, but whose effect is actually solely emulated by the hypervisor. Examples of such privileged instructions are accesses to peripheral registers. The implied performance issues may be mitigated by recurring to *hardware-assisted virtualization* [57] and/or to para-virtualization.

2.3.2. Hardware assisted virtualization

Hardware-assisted virtualization is a technique where the hard-

[1] Examples of Type 1 are VMWare ESX, WindRiver Hypervisor®, Xen, XtratuM, etc.
[2] Examples of Type 2 are the Kernel Based Virtual Machine (KVM), VMware Workstation, VirtualBox, Oracle VM Server, etc.

[3] <http://www.fentiss.com/en/products/xtratum.html>.
[4] <http://www.virtuallogix.com>.
[5] <https://sites.google.com/site/realtimexen/>.

ware provides additional features that speed-up the execution of VMs. For example, an additional layer for translation in hardware of virtualized to physical memory addresses enables unmodified guest operating systems to manipulate their page tables without trapping; this is provided by the former Extended Page Tables (EPT) [73] and later VT-x [72] technologies from Intel, as well as by the Nested Page Tables (NPT) [77] and AMD-v [78] technologies from AMD, among others. With these modifications, VMs can actually manipulate only virtual page tables, while the physical page tables are under the control of the hypervisor. Another remarkable example is the one of a physical network adapter capable of behaving as multiple logical ones, each with its own MAC address and send/receive queues, to be used by different VMs running on the same physical system (e.g., as allowed by the Virtual Machine Device Queues – VMDq – technology by Intel [74]). These are particularly effective when coupled with additional mechanisms for I/O MMU virtualization, allowing for direct and safe access to (virtualized) hardware from VMs, without hypervisor mediation for each access. For example, the Virtualization for Directed I/O from Intel [75] and IOMMU/Vi from AMD technologies realize this concept. The Single-Root I/O Virtualization (SR-IOV) technology from Intel [76] goes beyond network peripherals, allowing for extending the concept to any PCIe peripheral. For example, this is extremely useful for virtualizing access to (GP) GPU accelerators. Such hardware capabilities enable to host multiple VMs on the same hardware by largely reducing the need for intervention of the hypervisor, and for the use of software emulation techniques, when VMs need to access physical resources.

2.3.3. Para-virtualization

Para-virtualization is a technique by which the guest operating system is modified so as to be aware to be running within a VM. This results in avoiding the wasteful emulation of virtualized hardware. Rather, the modified kernel and drivers of the guest operating system are capable of performing direct calls to the hypervisor (a.k.a., *hypercalls*) whenever needed.[6] The evolution of hardware-assisted virtualization, coupled with para-virtualization techniques, allow virtualized applications nowadays to achieve an *average* performance only slightly below the native one. However, due to the increase in software and/or hardware complexity, the responsiveness of virtualized systems keeps suffering of *much more pronounced tail-latency problems* (i.e., a *much worse worst-case performance*) as compared to natively running software.

2.3.4. Operating System level virtualization

OS-level virtualization is a technique by which a single operating system gives to user-space software the illusion of multiple operating system instances, or *containers*, each one behaving as an independent operating system. For example, each operating system container has its own space of *process IDs* (PID), its own memory, (virtual) CPUs, private file system, etc. For example, LXC for Linux[7] and Jails for FreeBSD[8] are examples of OS-level virtualization. These can be thought of as the evolution of the old UNIX chroot concept [59].

2.3.5. Application-level virtualization

Application-level virtualization, or *process-level virtualization*, is a software technique allowing an application to be developed for, and run on top of, a virtual instruction set which is independent of the actual underlying hardware. A special software program is capable of interpreting at run-time such instructions, executing them on the real hardware. This is for example the case of Java applications, which are compiled to *bytecode* and need a Java Virtual Machine (JVM) at run-time for being run. A similar situation is the one of .NET applications. A recurrent optimization used at run-time to avoid the performance penalty due to interpretation of the virtual instructions, is the one of *just-in-time compilation* (JITC), in which the instructions are compiled into native instructions of the underlying *instruction set architecture* (ISA) on the fly. JITC improves greatly the average-case performance. However, it also introduces further predictability issues, namely the first execution of a code segment may be greatly slower than the subsequent executions.

2.3.6. Network virtualization

Tightly coupled with the recalled machine virtualization techniques are *network virtualization* techniques [35], which allow to emulate network set-ups in software. These are useful for example for allowing multiple VMs to run on the same physical host, with their own IP addresses, connected together in a virtual bridge topology. Another example, is the one of a set of VMs deployed on a set of physical hosts, which have all IP addresses belonging to the same sub-net/LAN as they belong to the same application or customer. However, the physical hosts that the VMs run within may be arranged in a completely independent topology; hosts may not even be deployed in the same LAN similarly to VPN set-ups. Network virtualization raises issues in predictability of hosted time-sensitive applications. Indeed, software components of an application running within different VMs may experience very short communication latencies when the VMs reside in the same physical host; the hypervisor is also designed to optimize such communication paths avoiding wasteful processing of packets within virtualized network stacks. However, they may also experiment very long communication latencies when the VMs actually reside in different hosts, different racks, different LANs or, in extreme cases, different data centres of the same cloud provider.

3. Real-time challenges in cloud computing

3.1. Terminology mapping

In essence, both real-time and cloud computing communities have the goal of guaranteeing the assigned resources or levels of contracted services between users and the execution platform or between VMs and the platform. However, what it means to a cloud computing person (typically coming from the fields of networking, distributed systems, or operating systems) may be somehow different from the idea conceived from a real-time systems point of view. Therefore, a clarification of concepts and terminology mapping is proposed in this section as shown in Table 2.

In real-time and embedded systems design, there is an increasing need for recurring to virtualization in order to enhance flexibility and isolation among independent run-time environments. It is needed to find direct ways to translate cloud *service level agreements* (SLAs) to resource assignment and enforcement at the platform level. SLAs typically deal with bandwidth, latency, memory, etc., that are in fact platform resources. However, a real-time perspective needs to step in the scene to provide actual execution mechanisms that guarantee and enforce execution (temporal and spatial) isolation to meet the timing requirements of real-time applications. Virtualization of the resources should remain at an equivalent distance from the application execution and platform level control over the computational resources. Providing an integrated resource management framework may be especially complicated in this domain specifically for resources as networks.

[6] The performance advantages of para-virtualization over full-virtualization have to face the weakened isolation from a security perspective, but these aspects are outside the scope of this paper.

[7] More information is available at: http://lxc.sourceforge.net/.

[8] More information is available at: http://www.freebsd.org/doc/handbook/jails.html.

Table 2
Terminology mapping for the domains of cloud computing and real-time.

Cloud terminology		Real-time concern	
Multi-tenancy	Avoidance of interference of multiple workloads in the same platform	Spatial and temporal isolation	Applications must not interfere either on the processor resource (to avoid deadline misses) nor on the usage of memory (to avoid synchronization delays that may cause deadline misses)
Dynamic provisioning	Autonomous configuration, allocation, and deployment of virtual cluster resources	Dynamic resource management	Resource managers are typically implemented close to (and even as part of) the real-time operating system. They include the needed policies to temporarily scale up/down resource assignments if it does not cause any temporal or spatial conflict
Service Level Agreements	Guaranteed QoS levels for applications that include the computational resources and networking performance.	Resource contracts and budget enforcement	Applications negotiate with the resource managers their required resource assignments. Resource managers have admission policies that are able to determine whether a specific assignment can be fulfilled. If the contract is set, the system will guarantee it at all times
QoS guarantees	Quality of service with respect to network performance parameters that guarantee the level of service for multiple users	Temporal guarantees	Timeliness of the execution is required in real-time systems. Results, i.e., either operations or communications, must fulfill specific deadlines. QoS is also considered as trading-off assigned resources for the quality of the output

In this line, one may think of the increasingly importance of network functions virtualization [61], including virtualized base stations [60]. As a consequence, there is an increasing need for controlling the temporal behavior of virtualized software, making their behavior more predictable. In other words, these is an increasing need for real-time virtualization techniques, enabling the existence of real-*time virtual machines* (RT-VMs) [48].

3.2. Control and access to the execution platform

From the general cloud computing perspective, RT-VMs are an enabling technology allowing virtualized applications to meet QoS constraints (in terms of, e.g., throughput, latency, memory, computational and networking capabilities) as stated in contractual agreements among customers and providers, formalized in proper SLAs. Nowadays cloud computing is still in an early development stage in which best-effort techniques dominate the panorama, with their difficulties in achieving predictable behaviors. As a consequence, real-time and QoS requirements on cloud applications have some laxity, and QoS in real SLAs is almost a myth. However, the situation is rapidly changing, and the full potential of cloud technologies will be unveiled only when real-time design methodologies will be widely deployed, so to achieve predictable execution of software in the cloud. This is expected to be pushed by increased and quickly evolving customers needs and expectations, with more and more interactive services available on-line.

The challenges for integrating real-time in virtualization are specially related to the role and design of a real-time hypervisor as sketched in Fig. 2. Following, the specific challenges of such a task are listed:

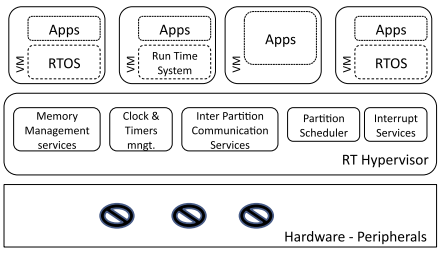

Fig. 2. Real-time virtualization view based on partitions. Hardware components are drawn as slashed circles. Software components are drawn as rounded boxes.

- Translation of interrupts is done by the hypervisor. Hardware interrupts are translated to software interrupts and forwarded to the operating system that must detect their priority. Once the interrupt is queued to be delivered to a VM, it will be up to the hypervisor scheduler to decide when to schedule the VM allowing it to handle its pending interrupt(s). In a moment in which there are more ready-to-run VM virtual CPUs than available physical CPUs, this may very well happen after tens of milliseconds.
- Access to time. Timers and clock should be readily available to applications that require to manage time. Also, guest real-time operating systems need it in order to perform the classical tasks of resource management such as resource accounting, enforcement and prediction [19,20,23,63]. However, the vision of time by guest operating systems may be largely affected by the schedule of VMs performed by the hypervisor, introducing further unpredictable behaviors.
- When dealing with multi-processor or multi-core VMs, the hypervisor might schedule the multiple virtualized CPUs (vCPUs) on the underlying physical CPUs in a way that does not ensure a uniform progress rate among the vCPUs. Furthermore, not all vCPUs are ensured to be scheduled at the same time. This may introduce further challenges that need to be handled in the software design; an example of this is the typical code segment within a guest operating system kernel in which a spin-lock operation is attempted, assuming the other vCPUs would free the lock in a few microseconds, while it has actually been scheduled out by the hypervisor.
- In distributed real-time virtualized environments, unpredictability of the network performance is one of the greatest challenges, especially in presence of virtualized networking set-ups. An interesting line of investigation is the one about tuning a hypervisor configuration so as to achieve certain performance goals for virtualized network function, e.g., as found in [83].

VMs must be scheduled for execution in a similar way that threads are scheduled by the operating system. VMs have scheduling attributes that are typically set statically and used by the hypervisor (or equivalent virtualization facilities) to schedule the execution of the VMs. For real-time scheduling, a RT-VM will set its attributes according to the real-time parameters needed or the QoS requirements so that it is scheduled with higher priority than others. In real-time scheduling theory, this can be combined with hierarchical scheduling algorithms/policies that define time partitions to execute sets of tasks that could be VMs. Some

standards like ARINC-653 [43] define temporal partitions for virtual machines that run in the same physical platform avoiding temporal and spatial interference.

In cloud computing for real-time embedded environments, the hypervisor is then in charge of managing the hardware resources and enforcing the spatial and temporal isolation of the guest operating systems. Para-virtualization is, therefore, the technique that better fits the requirements of embedded real-time systems. It provides faster execution and a simplified interface. The guest operating system must be customized in the para-virtualization case.

3.3. Real-time scheduling and resource management

Virtualization brings a number of challenging issues for real-time workloads. First, the increased level of resource sharing among multiple operating systems makes it difficult to run software in predictable ways, as the performance of each virtual machine (VM) depends on the amount of resources (e.g., computing, storage or networking) other VMs are consuming. What is worse, in a cloud computing environment multiple VMs sharing the same physical hosts and networks are often hosted on behalf of different and independent tenants (customers). Furthermore, dynamic creation and migration of VMs introduces potentially high and bursty work-loads that can greatly interfere with the performance of VMs sharing the same physical resources. This makes the problem of temporal interferences among VMs even more critical and important, as the performance exhibited by one VM does not only depend on the workload imposed within the cloud infrastructure by the same tenant, but also on the one imposed by VMs of other tenants. A situation that, albeit acceptable in an infantry stage of cloud computing, is not destined to last for too long, as users will gain confidence in the technology and evolve their still primitive requirements. We report below some of the most important contributions in the area of real-time scheduling and resource management for QoS-sensitive virtualized workloads.

3.3.1. Distributed resource management

Techniques and algorithms for real-time scheduling and resource management in general are able to provide temporal execution guarantees. As a means to introduce QoS-based resource management to trade-off execution quality by the assigned resources, a number of contributions on resource management architectures such as [19–21,63] appeared for centralized systems. Also, specific algorithms for real-time task management based on QoS levels were developed [23,24]. In these contributions, the existence of a resource broker that guarantees application resources and contracts lies clearly inside the operating system. For distributed environments, resource managers lie in a partially centralized/partially distributed mode [44]. All these ideas have to be modified to suit the virtualization technology in real-time domains. There is no centralized point for the location of the resource manager. Instead, it will have to rely on the principles of hierarchical scheduling [45] and the associated realizations as ARINC-653 in order to identify the architectural placement of the different modules of a distributed resource manager that is also distributed inside the node itself, as shown in Fig. 3.

Fig. 3 shows this transition to a more complex design in the architecture of resource management, i.e., schedulers placement. The scheduler of the host operating system will define and arbitrate the temporal partitions among the virtual machines (or partitions). In guest OSs, there will be a local scheduler that, in the case of real-time systems, must be synchronized in design with the host/global scheduler. The exact placement in actual implementations may differ according to various parameters, e.g., placement of the scheduling queues, access to the processor time which will depend on the type of hypervisor used, etc.

3.3.2. Real-time scheduling of virtual machines

The problem of performance isolation in cloud computing, and especially the one of controlling the interferences at the computing level, can be partially mitigated by using proper scheduling algorithms at the hypervisor or host operating system level.

Concerning the isolation of virtualized software on the computing level, [65] proposes to use an EDF-based scheduling algorithm for Linux on the host to schedule virtual machines (VMs). Unfortunately, the authors make use of a scheduler built into a dedicated user-space process (VSched), leading to unacceptable context switch overheads. Furthermore, VSched cannot properly guarantee temporal isolation in presence of a VM that blocks and unblocks, e.g., as due to I/O. [30] investigates the performance isolation of virtual machines, focusing on the exploitation of various scheduling policies available in the Xen hypervisor [9]. Furthermore, Dunlap proposed [29] various enhancements to the Xen credit scheduler in order to address various issues related to the temporal isolation and fairness among the CPU share dedicated to each VM. Precisely, [67] focuses on automatic on-line adaptation of the CPU allocation in order to maintain a stable performance of VMs. The framework needs to go beyond the common IaaS business model, in that it needs application-specific metrics to run the necessary QoS control loops. Also, the authors do not address how the dynamic resource allocation is considered from a resource planning and advance reservation perspective. In contrast to heuristics-based solutions discussed above, a promising approach is to employ rigorous real-time scheduling algorithms to schedule virtual machines. We provide an overview of two recent efforts in this direction in Section 4.

3.3.3. Alternatives not based on machine virtualization

A number of works advocated a disruptive approach truncating relationships with the nowadays practice of virtualized infrastructures for cloud computing, proposing completely alternative ways for enabling real-time performance in distributed cloud infrastructures. These are based on completely different software architecture conceptions. Indeed, various authors expressed the need for lighter software architectures supporting cloud applications, as compared to nowadays well-established virtualized infrastructures. In the current practice, it is easy to see replication of functionality at multiple levels, for example CPU scheduling, packet filtering/routing, memory allocation and security features at the hypervisor and guest operating system levels. This creates inertia in the management of the software components, impairing agility, adaptability and often performance.

For example, MediaCloud [28] was proposed as a novel framework specifically tied to multimedia processing. It is based on a notion of extremely lightweight and almost stateless media processing components whose location can be quickly decided and altered at run-time, for given multimedia flows among sources and destinations. For example, from a few measurements shown in [28], MediaCloud is capable to support instantiation of media processing functions, i.e., service components on distributed cloud resources, in the time-frame of a few milliseconds, and a

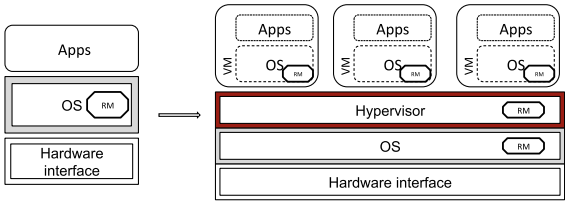

Fig. 3. Increased complexity in the architectural definition of resource managers (RMs).

re-assignment of media processing components from one processing resource onto another at service run-time in the time-frame of 2–3 ms. Such agility, achievable thanks to the specific focus of the platform which has been designed around the idea of an application-specific cloud, is surely a promising enabling factor for real-time workloads and adaptive resource management policies.

Similar motivations can be found in [41,84], where the novel Osprey operating system is proposed for predictable cloud computing, and in [85], where a novel operating system architecture for cloud computing is envisioned. Also, a similar line of reasoning can be found in the recently proposed Mirage [58] project, revisiting the idea of library operating systems for realizing high-performance network applications for cloud computing and mobile platforms with a highly reduced software stack optimized at compile-time. The reduced size of this type of cloud applications enable a dynamism and reactiveness in reconfiguring the system at run-time for dealing with performance issues that would simply be impossible with standard VMs.

Finally, in [50] it is highlighted that, in the hierarchical composition of multiple real-time schedulers, it is unavoidable to lose computing power in order to keep schedulability guarantees for the hosted real-time workloads. Alternatively, the authors suggest that OS-level virtualization, such as available through the Linux containers (LXC), may constitute valuable means to keep the most important advantages of virtualization (including isolation from a security perspective [70]), but with an easier and more efficient way to achieve schedulability of the overall real-time workload.

Various other contexts exist in which OS-level virtualization is being investigated, including network function virtualization and others, but their exhaustive description is out of the scope of the present paper.

As a final remark, it is worth to mention that some of the toughest challenges in cloud computing are due to the need for providing SaaS offerings with stable and predictable service levels [53,89,64], through the interaction of the multitude of business players that may be involved, including multiple SaaS and IaaS providers and Network Service Providers. The establishment of proper SLA models among these players may be a challenge on its own, encompassing technological as well as business aspects that deserve attention in the future.

3.4. Communication network challenges

The network is a source of temporal unpredictability in real-time systems. Real-time networking theoretical models calculate the response times of message exchanges and the overall system schedulability relying on the assumption that the network is highly predictable, no messages are lost, noise effects are not present, wired technology is used, and the system is closed so that no dynamic interference caused by extra tasks or applications can be considered once the system is in execution. Typical real-time networks for critical environments are based on the time-triggered architecture [42]. Such paradigm has been applied successfully in a number of domains, especially in automotive. In other application areas such as avionics, proprietary protocols have been used depending on the company standards and target system field (i.e., civil, military, or aerospace) such as the 1535 bus-based airborne data transfer systems.

In cloud computing environments, the network traffic and the nature of the networking protocols exhibits a specific behavior such that full predictability is not guaranteed. Instead QoS provisions are utilized.

3.4.1. Network QoS guarantees in HPC

Although deadline misses would not have catastrophic results in the main stream application domains of cloud computing, execution performance is a critical aspect for virtualization and cloud technology. Most cloud computing platforms do not specifically address real-time environments; however, there are a number of target systems with a critical requirements for service provisioning. These are called high performance cloud (HPC) applications. HPC applications have strong requirements for resource assignment guarantees and timely delivery of results. It is complex to meet such requirements in HPC applications over virtualized platforms since the overhead caused by I/O operations and processor cycles is very high. Moreover, activities are typically allocated to specific processors that often require thread run-time synchronization across cores. As a consequence, the main crucial aspects in HPC applications are:

- *Job allocation policies*. Jobs are frequently bound to specific cores.
- *Network connectivity of processor clusters*. Network protocols have to merge timeliness and throughput requirements to offer the needed performance levels.

To meet the different requirements of HPC in a virtualized environment, the virtualization platform must contain manager entities in charge of executing resource management policies that provide adequate trade offs for achieving the necessary performance levels. Different contributions such as [13] have identified network QoS as the primary problem for achieving HPC applications. Other contributions such as [14] have concluded that building high speed cluster interconnect networks as InfiniBand [15] in the virtual machines can bring in significant improvements. Network latencies can be significantly reduced as compared to other Ethernet based solutions. This approach is similar to the approaches delivered for distributed networked embedded systems such as the wormholes [17] though it is not focused on virtualization techniques.

For HPC, computing systems are usually built around high performance network interconnects that must somehow be integrated in the virtualized platform. To implement HPC systems, special network technologies are needed in order to guarantee a stable and constant level of QoS. Technologies such as Ethernet cause the network traffic to flow through the protocol stack of the guest and host operating system therefore decreasing performance.

To overcome the drawbacks of typical Ethernet for HPC, specific architectures for interconnection of cluster nodes are used that bypass the operating system to achieve low latency. Also, such technological approaches typically offload the protocol overheads to increase bandwidth explicitly executing some of the protocol layers inside the network board. Examples of such technologies are InfiniBand [15] and Myrinet [18]. Architectures such as InfiniBand do not get rid of the typical workings of protocol headers. However, they gain performance by more aggressive protocol processing; in fact, up to transport level is directly executed by the network interface. InfiniBand architecture, then, offers an communication-intensive environment where applications can directly use the network interface to transfer data without the intervention of the operating system. Applications access the memory of the device polling to determine the completion of their send/receive operations.

3.4.2. Virtualization of I/O network communications

As stated in [12], there are currently three approaches to the virtualization of I/O network communications that are suitable for HPC environments (Fig. 4):

- *PCI (peripheral component interconnect) passthrough*, where each VM in a node is granted direct access to a dedicated PCI I/O device/slot. As such, the number of VMs per host is

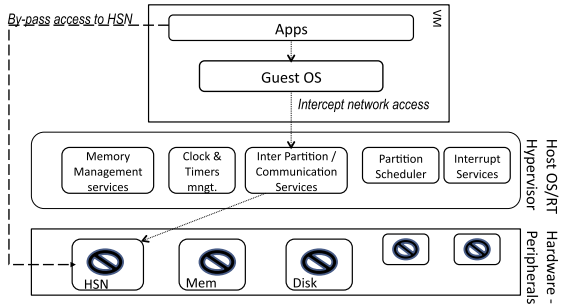

Fig. 4. OS by-passing for high performance networks/communications (HSN).

limited by the number of PCI slots since each one is used by an operating system. To provide spatial isolation between different VMs, an I/O MMU (memory mapping unit) is required.

- *Para-virtualization*, that is virtualization technique that presents a software interface to VMs that is similar but not identical to that of the underlying hardware. Therefore, some special operations are executed in the non-virtualized environment, and the operating system must be ported to run on the VM obtaining a more efficient and higher performance VM. Some approaches combining a high performance network architecture [16] over a para-virtualization technology hypervisor as Xen [9] have appeared that can achieve excellent performance.
- *Single root I/O virtualization (SR-IOV)*. It virtually multiplexes the PCI device offering each one as multiple devices called Virtual Functions (VFs). Here, a guest operating system can have exclusive access to VF by using PCI passthrough. SR-IOV is one of the most flexible options for some high performance networking architectures.

4. Example approaches

Following, we present only a subset of selected examples that illustrate some of the design decisions taken by virtualization technology to integrate real-time support.

4.1. Approaches to real-time virtual machine scheduling

Relating to Section 3.3.2, two significant lines of work dealt with real-time scheduling of virtual machines, namely IRMOS and RT-Xen, which are detailed in what follows.

4.1.1. IRMOS

In order to provide scheduling guarantees to individual VMs scheduled on the same system, processor and core, in the context of the IRMOS European project [27] a deadline-based real-time scheduler [99] for the Linux kernel was developed. It has been integrated within the Intelligent Service-Oriented Networking Infrastructure (ISONI) [37] software prototype developed throughout the project life-span, allowing for deployment of distributed virtualized applications with real-time performance guarantees. It provides temporal isolation among multiple possibly complex software components, such as entire VMs (with the KVM hypervisor, a VM runs as a Linux process). It uses a variation of the Constant Bandwidth Server (CBS) algorithm [71], based on Earliest Deadline First (EDF), for ensuring that each group of processes/threads is scheduled on the available CPUs for a specified time every VM-specific period, i.e., according to a resource reservation paradigm [62]. Interestingly, the same IRMOS scheduler may easily be reused for providing scheduling guarantees to LXC containers, as well as to JVM instances. Indeed, in all these cases, the scheduler allows for associating clearly specified scheduling guarantees to a set of processes running within the system. Additionally, the IRMOS RT scheduler can temporally isolate a VM from additional cloud management workload that is imposed onto the system by the cloud provider itself, like for example workload due to monitoring of the infrastructure, initial provisioning/deployment of VMs, or migration of VMs from one physical host to another. For example, assignment of precise resources reserved for VM migration, allows for enhancing the VM migration process with improved overall migration time and down-time [26,100]. The IRMOS approach has been validated on use-cases focusing on distributed real-time multimedia services in the cloud, including [27] e-Learning, high-performance video processing for film post-production and virtual and augmented reality for automotive. More information on the IRMOS real-time enhancements for the Linux kernel are provided in [99] and at the URL: http://lwn.net/Articles/398470/.

4.1.2. RT-Xen

The RT-Xen project has developed a real-time VM scheduling framework in the hypervisor. The RT-Xen scheduler [46] bridges the gap between real-time scheduling theory and the Xen platform by scheduling VMs using fixed-priority server algorithms designed based on real-time scheduling theory [90–92]. The real-time VM scheduler in the hypervisor and the schedulers in the guest operating systems form a scheduling hierarchy whose real-time schedulability can be formally analyzed using existing hierarchical real-time scheduling theory. Empirical evaluation showed that RT-Xen can provide effective real-time scheduling to guest Linux OSes at a 1 ms quantum, while incurring only moderate overhead for running the fixed-priority server algorithms [46]. This original RT-Xen 1.0 scheduler has been gone through two major enhancements in recent years. RT-Xen 1.1 supports compositional real-time scheduling where the resource demand of the tasks in a VM is represented by the VMs resource interface that can be computed based on compositional analysis [47]. If the resource interface is satisfied by the hypervisor scheduler, the guest operating system of the VM guarantees the schedulability of the tasks. RT-Xen 1.1 also incorporates improved periodic server algorithms that are work-conserving and improve task response times while preserving theoretical schedulability results. The effectiveness of RT-Xen 1.1 is demonstrated using workloads from an avionics case study (ARINC-653) [43]. The most recent version, RT-Xen 2.0, is a new real-time multi-core scheduler with a rich set of configurable features including global and partitioned schedulers, static and dynamic priority schemes, and different server algorithms [55].

With the increasing capacity of multi-core processors, there is an ongoing trend towards integrating multiple real-time systems as VMs co-located on a common host. It is important to properly prioritize the communication between VMs to meet their respective timing requirements. A key feature of the Xen architecture is that it relies on the manager domain to process packets between VMs. As a result both the VM scheduler and the manager domain can affect communication latency and incur priority inversion. A real-time VM scheduler alone cannot prevent priority inversion in inter-domain communication, as the manager domain can become the performance bottleneck in inter-domain communication. To address those limitations, RT-Xen provides the Real-Time Communication Architecture (RTCA) to support real-time communication between domains (VMs) co-located on a same physical host [55]. Experimental results demonstrate that RT-Xen can dramatically reduce the latency of high-priority communications between local VMs from milliseconds to microseconds through the combination of RTCA and the real-time VM scheduler. RTCA

has recently been extended to support real-time network communication between VMs on different physical hosts.

Several projects enhanced Xens default credit scheduler to better support soft real-time applications. For example, [56,95] provided a strict-prioritization patch for the credit scheduler so that real-time domains can always get resource before non real-time domains, and [96] mitigates priority inversion when guest domains are coscheduled with domain 0 on a same core. In [97], similar approaches are employed to improve the credit scheduler in the Xen ARM platform. While these works employed heuristics to enhance Xens existing credit scheduler, RT-Xen[9] provides a new real-time scheduling framework that is separate from the existing schedulers and is designed to deliver real-time performance based on compositional real-time scheduling theory.

4.1.3. Comparison between IRMOS and RT-Xen

RT-Xen and IRMOS share the common goal to enable predictable execution and real-time performance in virtualized environments. Both investigate the use of scheduling algorithms rigorously designed based on real-time scheduling theory to provide CPU scheduling guarantees for VMs, and both advocate the use of hierarchical scheduling techniques to assess the schedulability of real-time workloads running within the VMs.

Due to the architectural differences between Xen and KVM, while in RT-Xen such scheduling techniques have to be realized by modifying directly the scheduler of the Xen hypervisor, in IRMOS these changes are done at the level of the host operating system, namely changing the scheduler of the Linux kernel acting as host operating system [99]. Therefore, the IRMOS scheduler is useful for scheduling not only VMs, but also other real-time workloads such as web servers, Java virtual machines, or general real-time processes [48,49].

RT-Xen and IRMOS target at different application domains. IRMOS targets at mostly predictable execution for soft real-time, multimedia-oriented applications in virtualized cloud infrastructures, enabling a full real-time aware software life-cycle encompassing design, development, deployment, monitoring and run-time adaptation of real-time services. On the other hand, RT-Xen currently targets mainly predictable execution for both soft and hard real-time workloads for embedded, virtualized and networked embedded systems, with a focus on much lighter software infrastructures as required in many systems belonging to the core embedded domain. For example, to support many embedded systems with inter-VM dependencies RT-Xen enhances the real-time performance and predictability of the VM-to-VM communication within Xen [54] by mitigating priority inversion in the communication stack. Geared toward distributed cloud applications, IRMOS realizes an IaaS management scheme [34] allowing for combining various techniques for predictable execution and QoS support in the three core resources involved in distributed cloud applications, namely CPU scheduling, disk access and networking. This required to face a great number of challenges related to prediction of software performance, which have been tackled with the help of neural networks [33] and probabilistic modeling tools based on UML. The overall outcome of IRMOS has been a holistic solution and working prototype enabling deployment of distributed cloud applications with strong end-to-end QoS guarantees, either in a deterministic or in a probabilistic set-up, tackling the corresponding SLA-related challenges, both from a technological and from a business modeling viewpoint. Summarizing, IRMOS targets mostly predictable execution for soft real-time, multimedia-oriented applications in virtualized cloud infrastructures, enabling a full real-time aware software life-cycle encompassing design, development, deployment, monitoring and run-time adaptation of real-time services. On the other hand, RT-Xen targets mostly predictable execution for both soft and hard real-time workloads for embedded, virtualized and networked embedded systems, with a focus on much lighter software infrastructures as required in many systems belonging to the core embedded domain. With the increasing capacity of multi-core processors, there is an ongoing trend towards integrating multiple real-time systems as VMs co-located on a common host. It is needed to properly prioritize the communication between VMs to meet their respective timing requirements. A key feature of the Xen architecture is that it relies on the manager domain to process packets between VMs. As a result both the VM scheduler and the manager domain can affect communication latency and incurin priority inversion. A real-time VM scheduler alone cannot prevent priority inversion in inter-domain communication, as the manager domain can become the performance bottleneck in inter-domain communication. To address those limitations, the Real-Time Communication Architecture (RTCA) of RT-Xen supports real-time communication between domains (VMs) co-located on a same physical host [55]. Experimental results demonstrate that RT-Xen can dramatically reduce the latency of high-priority communications between local VMs from milliseconds to microseconds through the combination of RTCA and the real-time VM scheduler. RTCA has recently been extended to support real-time network communication between VMs on different physical hosts.

4.2. Approaches to real-time communication

Recent research has explored approaches to support real-time communication between VMs. In the following, we highlight the respective approaches employed in IRMOS, RT-Xen, and iLAND to support real-time communication.

Firstly, iLAND and ISONI are given as examples of contributions that improve the predictability of the network traffic scheduling. iLAND [44] offers a virtualized middleware based on DDS with enhanced functionality in the form of a VMM, without network bypassing. It provides a virtualized infrastructure where service-based applications can execute in both an isolated mode or ported to an open cloud computing setting. Applications based on iLAND services are flexible in the sense that they are able to change their structure, i.e., reconfigure according to a time-bounded protocol [24] respecting the specified timing constraints entered as SLAs. In iLAND, off-line fine-tuning of SLAs can be done with a priori performance analysis of virtualized DDS implementations (the communication backbone) [22,68]. Isolation of the traffic of independent VMs within ISONI [37] is achieved by a VSN individual virtual address space and by policing the network traffic of each deployed VSN. The two-layer address approach avoids unwanted crosstalk between services sharing physical network links. Mapping individual virtual links onto diverging network paths allows for a higher utilization of the network infrastructure by mixing only compatible traffic classes under similar predictability constraints and by allowing selection of more than just the shortest path. Traffic policing avoids that the network traffic going through the same network elements causes any overload leading to an uncontrolled growth of loss rate, delay and jitter for the network connections of other VSNs. Therefore, bandwidth policing is an essential building block to ensure QoS for the individual virtual links. It is important to highlight that ISONI allows for the specification of the networking requirements in terms of common and technology-neutral traffic characterization parameters, such as the needed guaranteed average and peak bandwidth, latency and jitter. An ISONI transport network adaptation layer abstracts from technology-specific QoS mechanisms of the networks, like Differentiated Services [25], Integrated Services [39,40] and MPLS [36].

[9] More information on RT-Xen, including documentation and open-source software, can be found at: https://sites.google.com/site/realtimexen/.

The specified VSN networking requirements are met by choosing the most appropriate transport network, among the available ones. Furthermore, it is not excluded that a network provider may be willing to deploy state-of-the art packet scheduling algorithms for either deterministic or probabilistic latency control, like found in [51,52], just to mention a few. More detailed information on QoS provisioning between data centers within an IaaS domain is given in [38]. Other interesting results from the research carried out in IRMOS include algorithms for the optimum placement of distributed virtualized applications with probabilistic end-to-end latency requirements [32], a probabilistic model for dealing with workload variations in elastic cloud services [79,93] and the use of neural networks for estimating the performance of virtual machines execution under different scheduling configurations [33]. The effectiveness of IRMOS/ISONI has been demonstrated for example through an e-Learning demonstrator [27].

In the domain of virtualized Telco (vTelco) services, latency-aware placement of vTelco applications has been investigated in [31], where application-level latency expressions have been introduced to account for the correct number of expected round-trip interactions among application components to be deployed. Also, in [80,81] an architecture for implementing a distributed Mobility Management Entity (dMME) in a scalable and latency-aware fashion has been presented.

5. Future directions

The research community is working on cloud computing from different perspectives, though still with a predominant distributed and networking side to it. Future directions in those areas are typically related to faster, more reliable, and higher (guaranteed) bandwidth networking for data bulks; security and privacy guarantees on data centers and user links; more efficient data access and analysis mechanisms and algorithms. From other domains, new business models are also being engineered and thought out to get the most out of this environment.

First we provide some ideas on one of the key problems in cloud computing such as the improvement of the virtualization technology to offer strict guarantees over the contracted resources. Later, we focus on two of the key challenges of cloud computing as it is the support for bulk data transfer, analysis, and storage; and virtualization of network functions.

5.1. Virtualization technology

From the distributed real-time perspective, the essential target is the improvement of the virtualization technology to guarantee contracted resources at the level of the execution platform in order to meet the applications timing properties. To achieve such goal, there are different directions of interest to drive towards. On one side, hardware support for virtualized computing has been moving towards improving the throughput in the average case, rather than allowing for more predictable execution. Similarly, design of operating systems for general-purpose computing is geared towards average performance rather than predictability. In para-virtualization it is a challenge to port existing operating systems to run in some virtual machine schedulers. Hardware vendors are realizing that virtualization is becoming more and more important, particularly in data centres. Other important directions are the design of scheduling and execution algorithms that enable, in the same VM, *job co-existence with temporal and spatial isolation guarantees*. Albeit the real-time research community addressed hierarchical scheduling from a theoretical standpoint, the concrete support of these concepts within virtualized environments is practically null, leaving developers and practitioners clueless about the way to possibly achieve temporal isolation and performance guarantees for critical tasks within guest operating systems. Support for *dynamic job scheduling* would enable any new job to join a system in execution without altering the temporal properties of the whole system. This requires to implement multi-level resource managers *integrating general purpose network protocols* as they are the mainstream in cloud, e.g., Ethernet. One other challenge will be the real-time live migration where VMs containing real-time applications are transferred between different physical servers.

5.2. Support for data-intensive and Big-Data workloads

One of the increasingly interesting applications of cloud computing is in the context of management of massive amounts of data, such as data analytics and real-time processing engines in the area of Big-Data. Indeed, such applications are difficult if not impossible to run within traditional local premises, while when running in the cloud, it is easy to take advantage of the big storage capacity as well as of the computational capabilities that are available within a data centre. Differently from applications traditionally considered in the context of real-time systems, data-intensive workloads exercise an unusual stress on the memory sub-system of a computing system, as well as on its networking capabilities, in addition to the pure CPU computing load.

Classical techniques for temporal isolation of computing workloads, based on CPU scheduling, and particularly on real-time scheduling and resource reservations, such as the IRMOS scheduler or the RT-Xen approaches presented above, struggle at guaranteeing the desirable level of performance isolation in presence of data-intensive workloads. Indeed, various open challenges arise in this context:

- Even though VMs get precise guarantees by the hypervisor CPU scheduler, their actual performance keeps being greatly variable as due to the *interferences happening on the memory sub-system level*; indeed, whenever scheduling in and out VMs on a CPU core, the cache(s) may easily be wiped out when the co-scheduled workload is data-intensive; even though VMs are deployed and pinned down onto separate cores, the 3rd-level and sometimes the 2nd-level caches are shared with other cores, thus with other VMs, leading to big unpredictable interferences that are challenging to be modeled and accounted for; on big multi-core and NUMA machines, even with separate caches, data-intensive workloads create massive amounts of cache misses, thus requests to load cache lines from the memory controllers, which are normally shared across a number of cores and physical processors, creating interferences on the level of access to the memory controller(s).
- The VMs hosted on a big multi-core machine may need to share access to the same physical networking hardware; when the amounts of data to be transferred by each VM are massive, even with 10 Gb/s Ethernet interfaces, it is challenging to design a system that guarantees given temporal properties; for example, the computing platform is flooded with interrupts due to the network adapter in a way that is uncommon and affects workloads in a fairly unpredictable way; imagine a CPU-bound workload co-hosted with a data-bound workload, where interrupts generated by the latter impact on the performance of the former.

Clearly, from a real-time perspective, the above problems might be addressed by proper worst-case analysis techniques that properly account for the mentioned types of interference. These types of techniques might be effective on embedded systems with a relatively known set-up and control on the hosted software; however, on virtualized big multi-core environments hosting multi-tenant virtualized systems that are completely independent and have

(possibly) very heterogeneous characteristics, worst-case techniques are far from being usable; they would yield a level of under-utilization of the resources that would kill the whole concept of Cloud Computing.

Interestingly, a few approaches that move along the lines of trying to mitigate the above mentioned problems in presence of data-intensive workloads have been recently proposed in the real-time research literature, such as [66].

5.3. Real-time challenges in network functions virtualization

One of the disruptive technologies that is emerging in the area of Cloud Computing and data centres architectures is the one of Network Functions Virtualization (NFV) [61]. In cloud infrastructures, the hosted workloads present increasingly complex networking demands, and the providers also have increasing needs for flexibility in the management of the underlying infrastructure. Flexibility is achievable by reallocation of VMs, applications, and data storages as needed by the run-time status of the infrastructure. Nevertheless, such flexibility must preserve a fixed (and agreed-upon) logical view of a virtualized networked environment with specific QoS requirements and SLAs. All this requires complex management and an increasing use of software-based approaches, including highly flexible and dynamic virtualization and network virtualization techniques. Network virtualization proposes decoupling of functionalities in a networking environment by separating the role of the traditional Internet Service Providers (ISPs) into two: infrastructure providers that manage the physical infrastructure, and service providers that create virtual networks by aggregating resources from multiple infrastructure providers and offer end-to-end network services. In this context, it is interesting to see how a hypervisor configuration may impact on the performance of virtualized network functions [83], and how such configurations could be integrated within an automated framework for QoS-aware management of a cloud infrastructure.

Network virtualization for cloud computing is focused at the different networking layers (layer 3, 2, or 1). For example, Layer 3 VPNs or Virtual Private Networks (L3VPN) are distinguished by their use of layer 3 protocols (e.g., IP or MPLS) in the VPN backbone to carry data between the distributed end points with possible heterogeneous tunneling techniques. Quality of service with respect to network performance has to be guaranteed even when multiple users are sharing a specific infrastructure simultaneously. Some studies (such as [82]) have shown that performance can improve significantly if virtual machines are interoperated via a high-speed cluster interconnect. There are some implementations (such as [12]) that back up this idea based on the usage of InfiniBand [15], providing improved networked latencies as compared to IaaS solutions based on Ethernet. However, merging networking research similar to the one exposed above with real-time design and scheduling techniques is an open area of research. Typically, passed the medium access layer that controls deterministic contention and transmission, networking is typically not the focal point of real-time research but rather of high performance networking. As a consequence, network virtualization merged with cloud technology in real-time environments is an extremely important, challenging, and open area of research.

Additionally, network equipment manufacturers and network providers are exploring the additional virtualization techniques to make networks more flexible and adaptable to a number of novel and emerging scenarios, which also enable novel business models for the networking industry. This is the case, for example, of future virtualized base stations [60] in which the same hardware may be shared among a number of providers, as opposed to the current practice in which every provider deploys its own physical infrastructure.

Similarly, there is a general trend towards providing network functions that used to be available as separate physical equipment. Network functions have the form of software components and services that can be deployed on standard general-purpose computing hardware, or sometimes even in the cloud. This is the case for example of security services or routing logic such as OpenFlow controllers for Software-Defined Networks (SDNs) [69]. This will have a number of benefits such as significant reduction of equipment, energy consumption, or maintenance tasks, among others. Further investigation is also needed on the side of how to properly *distribute* such functions over the network, in order to maintain scalability of the infrastructure, following up on the architecture proposed in [81] for cellular network functions, for example.

A last discussion in also needed for Ethernet technology as it occupies a predominant position in local area networks and, consequently, also in mainstream cloud computing data centres. Ethernet presents problems of contention in the access to the physical communication media. Over the last decades, this has led to a myriad of research works targeted at deriving QoS performance metrics for estimating suitability of different actual deployments and providing priority assignment techniques for different applications, users, or data flows, or to guarantee different levels of performance to communication data flows. Quantitative measures of QoS have focused on aspects of the network service such as transmission error rates, bandwidth, throughput, transmission delay, availability, jitter, packet loss/drop probability, etc. Ethernet has also been used in real-time settings with the necessary precautions to reduce (and even avoid) the contention over the communication media at the cost of significantly reducing the average and peak throughput and bandwidth. Most solutions have included the usage of switched Ethernet. However, such a design eliminates the contention from the communication media by transferring it to the router queues, and it also requires the global scheduling and synchronization of the network nodes. Real-time cloud computing applications such as on-line gaming and on-line video streaming highly demand network resources since they often require fixed bit rate and are delay sensitive. QoS guarantees are very important to them if the network capacity is insufficient. In addition to the research work on improving the utilization of the network bandwidth and fine-grain characterization of the performance metrics of different protocols, other alternatives that augment the bandwidth of Ethernet, such as *x*-Gigabit solutions, will continue to rise in the near future in order to better support the increasing density of VMs and processor cores in cloud computing data centres.

6. Conclusion

Although it is a reality already for some years, cloud computing is a fairly new paradigm for dynamically provisioning computing services located in data centers that are intensive in the use of virtualization technology allowing server consolidation and efficient resource usage in general. In the last decades, important advances mainly at machine virtualization, network, data analysis, and storage levels have contributed to the wide spread usage and adoption of this paradigms in different domains.

Due to their strong timing requirements and needed predictability guarantees, real-time application domains are still far behind in the full adoption of cloud computing. Merging cloud computing with real-time is a complex problem that requires real-time virtualization technology to consolidate the predictability characteristics that it will be able to offer in the future. Currently, there are still some barriers and important challenges for its full adoption by real-time domains and especially hard real-time applications. This is mainly due to the fact that predictability

is challenged by the limited capacities offered to real-time guest software to access to the hardware resources; the communication latencies in distributed nodes are challenged by the Internet protocols that dominate the sector and that are executed in a heavy/thick software stack; the security of cloud and virtualization deployments; and the need for introducing hierarchical schedulers capable of providing temporal guarantees. Although there are some solutions that are more advanced for embedded systems and also provide some real-time capabilities such as Xen, still available commercial solutions do not guarantee execution isolation with a real-time software stack in a cloud computing environment. This paper has analyzed some of the problems and challenges for achieving real-time cloud computing as a first step towards presenting an abstract map of the situation today, identifying the needed elements at all levels to make it happen. The presented concerns range from the hypervisor structure and role, the different possible types of virtualization and their respective performance, general resource management concerns and schedulability of VMs related to hierarchical scheduling, and the important role of the network in the overall picture of virtualization technology. For the latter, this paper has described some solutions of the HPCC community to bypass the bottlenecks introduced by the protocol execution inside the different layers of the software stack. A terminology mapping between cloud and distributed real-time systems domains has been settled in order to connect both areas. Lastly, we have introduced a number of future directions that require attention from different communities to realize the real-time cloud.

Acknowledgements

This research was supported, in part, by the Salvador de Madariaga Programme for International Research Stays funded by the Spanish Ministry of Education under contract PRX12/00252; by the Spanish National Project REM4VSS (TIN2011-28339); by the US ONR award N000141310800; and by US NSF grant CNS-1329861 (CPS).

References

[1] J.E. Smith, R. Nair, The architecture of virtual machines, IEEE Comput. (2005).
[2] M. Claypool, K. Claypool, Latency and player actions in online games, Commun. ACM 49 (11) (2006) 40–45.
[3] C.Y. Huang, C.H. Hsu, Y.C. Chang, K.T. Chen, Gaming Anywhere: an open cloud gaming system, in: Proc. of 4th ACM Multimedia Systems Conference, 2013, pp. 36–47.
[4] R. Shea, J. Liu, E.H. Ngai, Y. Cui, Cloud gaming: architecture and performance, IEEE Netw. 27 (4) (2004).
[5] IBM Corporation, IBM Systems: Virtualization. v2, release 1. http://publib.boulder.ibm.com/infocenter/eserver/v1r2/topic/eicay/eicay.pdf, 2006
[6] M. Masmano, I. Ripoll, A. Crespo, XtratuM: a Hypervisor for Safety Critical Embedded Systems, in: Proc. of 11th Real-Time Linux Workshop, Dresden, Germany, 2010.
[7] A. Crespo, I. Ripoll, M. Masmano, Partitioned embedded architecture based on hypervisor: the XtratuM approach, in: Proc. of 8th European Dependable Computing Conference (EDCC-8), Valencia, Spain, April 2010.
[8] SCOPE Alliance, Virtualization: State of the Art, Version 1.0 http://www.scope-alliance.org/sites/default/files/documents/SCOPE-Virtualization-StateofTheArt-Version-1.0.pdf, 2008
[9] P. Barham et al., Xen and the art of virtualization, in: Proc. of the ACM Symposium on Operating Systems Principles, 2003, pp. 164–177.
[10] W. Huang, J. Liu, B. Abali, D. Panda, A case for high performance computing with virtual machines, in: Proc. of 20th Annual ACM Int'l Conference on Supercomputing, 2006, pp. 125–134.
[11] M. Mergen, V. Uhlig, O. Krieger, J. Xenidis, Virtualization for high-performance computing, in: Proc. of ACM SIGOPS Operating Systems Review, vol. 40 (2), 2006, pp. 8–11.
[12] V. Mauch, M. Kunze, M. Hillenbrand, High performance cloud computing, Future Generat. Comput. Syst. 29 (6) (2013) 1408–1416.
[13] A. Gupta, D. Milojicic, Evaluation of HPC applications on cloud Technical report, HPL-2011-132, HP Laboratories, 2011.
[14] M. Hillenbrand, V. Mausch, J. Stoess, K. Miller, F. Bellosa, Virtual InfiniBand Cluster for HPC clouds, in: 2nd Int'l Workshop on Cloud Computing Platforms, NY, USA, December 2012.
[15] InfiniBand Trade Association, InfiniBand Architecture specification, vol. 1, release 1.2.1., 2007.
[16] J. Liu, W. Huang, B. Abali, D. Panda, High performance VMM-bypass I/O in virtual machines, in: Proc. of USENIX Conference, 2006.
[17] P. Verissimo, Traveling through wormholes: a new look at distributed systems models, ACM SIGACT, 2006.
[18] Myrinet, http://www.myricom.com/scs/myrinet/overview/, 2013
[19] C. Otero, L. Steffens, P. van der Stok, S. van Loo, A. Alonso, J. Ruiz, R. Bril, M. García Valls, Ambient Intelligence: Impact on Embedded Systems Design, Chapter On: QoS-Based Resource Management for Ambient Intelligence, Kluwer Academic Publishers, 2003.
[20] M. García-Valls, A. Alonso Muñoz, J. Ruíz Martínez, A. Groba, An architecture of a quality of service resource manager for flexible multimedia embedded systems, in: Proc. of 3rd International Workshop on Software Engineering and Middleware (SEM2002), In Lecture Notes in Computer Science, vol. 2596, 2003.
[21] L. Palopoli, T. Cucinotta, L. Marzario, G. Lipari, AQuoSA – adaptive quality of service architecture, Softw. Pract. Exp. 39 (2009) 1–31.
[22] M. García Valls, P. Basanta-Val, R. Serrano-Torres, Benchmarking communication middleware for cloud computing virtualizers, in: Proc. of 2nd International Workshop on Real-Time and Distributed Computing in Emerging Applications (REACTION 2013). Vancouver, Canada, December 2013, pp. 13–18.
[23] M. García Valls, A. Alonso, J.A. de la Puente, A dual-band priority assignment algorithm for dynamic QoS resource management, Future Gener. Comp. Sy. 28 (2012) 902–912.
[24] M. García-Valls, P. Uriol-Resuela, F. Ibáñez-Vázquez, P. Basanta-Val, Low complexity reconfiguration for real-time data-intensive service-oriented applications, Future Generat. Comput. Syst. 37 (2014) 191–200.
[25] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, RFC2475, an architecture for differentiated service, IETF, 1998.
[26] F. Checconi, T. Cucinotta, D. Faggioli, G. Lipari, Hierarchical multiprocessor CPU reservations for the Linux kernel, in: Proceedings of the 5th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT 2009), Dublin, Ireland, June 2009.
[27] T. Cucinotta, F. Checconi, G. Kousiouris, K. Konstanteli, S. Gogouvitis, D. Kyriazis, T. Varvarigou, A. Mazzetti, Z. Zlatev, J. Papay, M. Boniface, S. Berger, D. Lamp, T. Voith, M. Stein, Virtualised e-learning on the IRMOS real-time cloud, Service Orient. Comput. Appl. (2011). doi: 10.1007/s11761-011-0089-4 (pages 116).
[28] P. Domschitz, M. Bauer, Mediacloud – a framework for real-time media processing in the network, in: Proceedings of EuroView 2012, Wuerzburg, Germany, July 2012.
[29] G. Dunlap, Scheduler development update, Xen Summit Asia, Shanghai, 2009.
[30] D. Gupta, L. Cherkasova, R. Gardner, A. Vahdat, Enforcing performance isolation across virtual machines in Xen, in: Proceedings of the ACM/IFIP/USENIX International Conference on Middleware, Springer-Verlag New York Inc., 2006 (pages 342362).
[31] F. Chang, R. Viswanathan, T.L. Wood, Placement in clouds for application-level latency requirements, in: Proceedings of the 5th IEEE International Conference Cloud Computing (IEEE CLOUD 2012), Honolulu, Hawaii, USA, June 2012.
[32] K. Konstanteli, T. Cucinotta, T. Varvarigou, Optimum allocation of distributed service workflows with probabilistic real-time guarantees, Service Orient. Comput. Appl. (68) (2010) 22968:243.
[33] G. Kousiouris, T. Cucinotta, T. Varvarigou, The effects of scheduling, workload type and consolidation scenarios on virtual machine performance and their prediction through optimized artificial neural networks, J. Syst. Softw. (2011).
[34] D. Kyriazis, A. Menychtas, G. Kousiouris, K. Oberle, T. Voith, M. Boniface, E. Oliveros, T. Cucinotta, S. Berger. A Real-time Service Oriented Infrastructure. in Proceedings of the Annual International Conference on Real-Time and Embedded Systems (RTES 2010), Singapore, November 2010.
[35] K. Oberle, M. Kessler, M. Stein, T. Voith, D. Lamp, S. Berger, Network virtualization: the missing piece, in: 13th International Conference on Intelligence in Next Generation Networks (ICIN 2009), October 2009, pp. 1–6.
[36] E. Rosen, A. Viswanathan, R. Callon, RFC3031, Multi-protocol Label Switching Architecture, IETF, January 2001.
[37] T. Voith, M. Kessler, K. Oberle, D. Lamp, A. Cuevas, P. Mandic, A. Reifert, ISONI Whitepaper v2.0., 2009.
[38] T. Voith, K. Oberle, M. Stein, Quality of service provisioning for distributed data center inter-connectivity enabled by network virtualization, Elsevier Future Generation Computer Systems (FGCS 2011), 2011.
[39] J. Wroclawski, RFC 2210, The Use of RSVP with IETF Integrated Services, IETF, September 1997.
[40] J. Wroclawski, RFC2211, Specification of the Controlled Load Quality of Service, IETF, September 1997.
[41] J. Sacha, J. Napper, S. Mullender, J. McKie, Osprey: Operating system for predictable clouds, Dependable Systems and Networks Workshops (DSN-W), 2012 IEEE/IFIP 42nd International Conference on, pp. 1–6, 25–28 June 2012.
[42] H. Kopetz, G. Bauer, The time-triggered architecture, in: Proceedings of the IEEE, vol. 91(1), January 2003.
[43] Aeronautical Radio, Inc., Avionics Application Software Standard Interface: ARINC Specification 653P1-3, 2010, pp. 11–15.
[44] M. García Valls, I. Rodríguez López, L. Fernández Villar, iLAND: an enhanced middleware for real-time reconfiguration of service oriented distributed real-time systems, IEEE Trans. Ind. Inf. 9 (1) (2013) 228–236.

[45] Z. Deng, J.W. Liu, Scheduling real-time applications in an open environment. IEEE Real-Time Systems Symposium, 1998.
[46] S. Xi, J. Wilson, C. Lu, C.D. Gill, RT-Xen: Towards Real-time Hypervisor Scheduling in Xen ACM International Conference on Embedded Software (EMSOFT), October 2011.
[47] J. Lee, S. Xi, S. Chen, L.T.X. Phan, C. Gill, I. Lee, C. Lu, O. Sokolsky, Realizing Compositional Scheduling through Virtualization IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), April 2012.
[48] T. Cucinotta, G. Anastasi, L. Abeni, Real-Time Virtual Machines In Proceedings of the 29th Real-Time System Symposium (RTSS 2008). Work in Progress Session, Barcelona, December 2008.
[49] T. Cucinotta, G. Anastasi, L. Abeni, Respecting temporal constraints in virtualised services, in: Proceedings of the 2nd IEEE International Workshop on Real-Time Service-Oriented Architecture and Applications (RTSOAA 2009), Seattle, WA, July 2009.
[50] L. Abeni, T. Cucinotta, Efficient virtualisation of real-time activities, in: Proceedings of the IEEE International Workshop on Real-Time Service-Oriented Architecture and Applications (RTSOAA 2011), December 12–14 2011, Irvine, CA.
[51] M. Andrews, L. Zhang, Minimizing end-to-end delay in high-speed networks with a simple coordinated schedule, J. Algorithms 52 (1) (2004) 57–81.
[52] M. Andrews, L. Zhang, Satisfying arbitrary delay requirements in multihop networks, in: Proc. of the 27th IEEE Conference on Computer Communications (INFOCOM 2008), 13–18 April 2008, pp. 116–120.
[53] T. Cucinotta, S. Gogouvitis, K. Konstanteli. SLAs in virtualized cloud computing infrastructures with QoS assurance, in: Proceedings of the International Workshop on eContracting in the Clouds, co-located with the eChallenges 2011 Conference, Florence, Italy, October 27 2011.
[54] S. Xi, C. Li, C. Lu, C. Gill, Prioritizing local inter-domain communication in Xen, in: Proc. of ACM/IEEE International Symposium on Quality of Service (IWQoS'13), June 2013.
[55] S. Xi, M. Xu, C. Lu, L. Phan, C. Gill, O. Sokolsky, I. Lee, Real-time multi-core virtual machine scheduling in Xen, in: Proc. of ACM International Conference on Embedded Software (EMSOFT'14), October 2014.
[56] M. Lee, A.S. Krishnakumar, P. Krishnan, N. Singh, S. Yajnik. Supporting soft real-time tasks in the Xen hypervisor. In VEE'10. Pittsburg, Pennsylvania, March 2010.
[57] D. Abramson, J. Jackson, S. Muthrasanallur, G. Neiger, G. Regnier, R. Sankaran, I. Schoinas, R. Uhlig, B. Vembu, J. Wiegert, Intel virtualization technology for directed I/O, Intel Technol. J. 10 (3) (2006).
[58] A. Madhavapeddy, R. Mortier1, C. Rotsos, D. Scott, B. Singh, T. Gazagnaire, S. Smith, S. Hand, J. Crowcroft, Unikernels: library operating systems for the cloud, in: Proceedings of ASPLOS'13, March 1620, 2013, Houston, TX, USA.
[59] M. Meierjohann, UNIX: Change-root environments for web applications. SANS GIAC LevelOne Security Essentials, March 30, 2001.
[60] P. Bosch, A. Brusilovsky, R. McLellan, S. Mullender, P. Polakos, Secure Base Stations Bell Labs Tech. J. 13 (4) (2009) 227–243.
[61] M. Chiosi, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Deng, J. Benitez, U. Michel, H. Damker, K. Ogaki, T. Matsuzaki, M. Fukui, K. Shimano, D. Delisle, Q. Loudier, C. Kolias, I. Guardini, E. Demaria, R. Minerva, A. Manzalini, D. Lpez, F. Javier Ramn Salguero, F. Ruhl, P. Sen, Network functions virtualisation. An introduction, benefits, enablers, challenges and call for action. SDN and OpenFlow World Congress, Darmstadt-Germany, October 22–24, 2012.
[62] C.W. Mercer, S. Savage, H. Tokuda, Processor capacity reserves: operating system support for multimedia applications, in: Proceedings of the International Conference on Multimedia Computing and Systems, pp. 90–99, 1994.
[63] T. Cucinotta, L. Palopoli, L. Abeni, D. Faggioli, G. Lipari, On the integration of application level and resource level QoS control for real-time applications, IEEE Trans. Ind. Inf. 6 (4) (November 2010).
[64] T. Cucinotta, D. Lugones, D. Cherubini, K. Oberle, Brokering SLAs for end-to-end QoS in Cloud Computing, in: Proceedings of the 4th International Conference on Cloud Computing and Services Science (CLOSER 2014), April 3–5, 2014, Barcelona, Spain.
[65] B. Lin, P. Linda, Vsched: mixing batch and interactive virtual machines using periodic real-time scheduling, in: Proceedings of ACM/IEEE Supercomputing, 2005.
[66] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo, Lui Sha, Memory access control in multiprocessor for real-time systems with mixed criticality, in: Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on, July 2012, pp. 299–308.
[67] R. Nathuji, A. Kansal, A. Ghaffarkhah. Q-clouds: managing performance interference effects for QoS-aware clouds, EuroSys, 2010, pp. 237–250.
[68] R. Serrano-Torres, M. García-Valls, P. Basanta-Val, Virtualizing DDS middleware: performance challenges and measurements, in: IEEE International Conference on Industrial Informatics (INDIN'13). Bochum, Germany, July 2013.
[69] IBM Systems and Technology, Software Defined Networking A new paradigm for virtual, dynamic, flexible networking, 2012.
[70] F. Bazargan, C.Y. Yeun, M.J. Zemerly, State of the art of virtualization, its security threats and deployment models, Int. J. Inf. Secur. Res. 2 (3/4) (2012).
[71] L. Abeni and G. Buttazzo Integrating Multimedia Applications in Hard Real-Time Systems. Proceedings of the 19th IEEE Real-Time Systems Symposium (RTSS 1998), Madrid, Spain, pp. 4–13, December 1998.
[72] Intel Corp, Hardware-Assisted Virtualization Technology, available at: http://www.intel.com/content/www/us/en/virtualization/virtualization-technology/hardware-assist-virtualization-technology.html
[73] Matthew Gillespie Best Practices for Paravirtualization Enhancements from Intel(R) Virtualization Technology: EPT and VT-d June 2009, available at: http://software.intel.com/en-us/articles/best-practices-for-paravirtualization-enhancements-from-intel-virtualization-technology-ept-and-vt-d?wapkw=ept.
[74] Shefali Chinni, Radhakrishna Hiremane, Intel Whitepaper: Virtual Machine Device Queues, 2007.
[75] David Ott, Understanding VT-d: Intel Virtualization Technology for Directed I/O, June 2009.
[76] Intel LAN Access Division PCI-SIG SR-IOV Primer: An Introduction to SR-IOV Technology – Revision 2.5, 2011.
[77] Advanced Micro Devices, Inc., Whitepaper: AMD-V(TM) Nested Paging, 2008.
[78] AMD, AMD Virtualization, available at: http://www.amd.com/us/solutions/servers/virtualization/Pages/virtualization.aspx.
[79] K. Konstanteli, T. Cucinotta, K. Psychas, T. Varvarigou, Admission control for elastic cloud services, in: 5th IEEE International Conference on Cloud Computing (IEEE CLOUD 2013), 2012, pp. 41–48.
[80] Xueli An, Fabio Pianese, Indra Widjaja, Utku Gnay, Acer MME: virtualizing LTE mobility management, in: 36th Annual IEEE Conference on Local Computer Networks (LCN 2011), Bonn, October 2011.
[81] X. An, F. Pianese, I. Widjaja, U.G. Acer, DMME: a distributed LTE mobility management entity, Bell Labs Tech. J. Spec. Issue Commun. Comput. Cloud 17 (2) (2012).
[82] N. Regola, J.C. Ducom, Recommendations for virtualization technologies in high performance computing, in: 2nd IEEE International Conference on Cloud Computing Technology and Science, 2010.
[83] L. Abeni, C. Kiraly, N. Li, A. Bianco, Tuning KVM to enhance virtual routing performance, in: Proceedings of the IEEE International Conference on Communications (ICC13), Budapest, Hungary, 2013.
[84] S.J. Mullender, Predictable cloud computing, Bell Labs Tech. J. Spec. Issue Commun. Comput. Cloud 17 (2) (2012) 25–39.
[85] F. Pianese, P. Bosch, A. Duminuco, N. Janssens, T. Stathopoulos, M. Steiner, Toward a cloud operating system, in: IEEE/IFIP Network Operations and Management Symposium Workshops (NOMS), April 2010, Osaka.
[86] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware allocation heuristics for efficient management of data centers for cloud computing, Future Generat. Comput. Syst. 28 (2012) 755–768.
[87] P. Mell, T. Grance, NIST SP800-145: The NIST Definition of Cloud Computing, September 2011.
[88] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger and D. Leaf, NIST Special Publication 500–292 – NIST Cloud Computing Reference Architecture, September 2011.
[89] K. Oberle, D. Cherubini, T. Cucinotta, End-to-end service quality for cloud applications, in: Proceedings of the 10th International Conference on Economics of Grids, Clouds, Systems and Services (GECON 2013), September 18–20, 2013, Zaragoza, Spain.
[90] L. Sha, J. Lehoczky, R. Rajkumar, Solutions for some practical problems in prioritized preemptive scheduling, in: RTSS, 1986.
[91] B. Sprunt, Aperiodic Task scheduling for real-time systems. PhD thesis, 1990.
[92] J. Strosnider, J. Lehoczky, L. Sha, The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments, IEEE Trans. Comput. 44 (1) (1995) 73–91.
[93] K. Konstanteli, T. Cucinotta, K. Psychas, T. Varvarigou, Elastic admission control for federated cloud services, IEEE Transactions on Cloud Computing, 2014.
[94] M. Chiosi et al. Network Functions Virtualisation – Introductory White Paper. SDN and OpenFlow World Congress. October 22–24, 2012, Darmstadt, Germany, available at: http://portal.etsi.org/NFV/NFVWhitePaper.pdf.
[95] W. Zhang, S. Rajasekaran, T. Wood, M. Zhu, MIMP: deadline and interference aware scheduling of Hadoop virtual machines, in: IEEE/ACM International Symposium on Cluster Cloud and Grid Computing, 2014.
[96] S. Govidan, A.R. Nath, A. Das, B. Urgaonkar, A. Sivasubramaniam, Xen and Co.: communication-aware CPU scheduling for consolidated Xen-based hosting platforms, in: VEE, 2007.
[97] S. Yoo, K.-H. Kwak, J.-H. Jo, C. Yoo, Toward under-millisecond I/O latency in Xen-ARM, in: Proceedings of the Second Asia-Pacific Workshop on Systems, ACM, 2011.
[98] Z. Gu, Q. Zhao, A state-of-the-art survey on real time issue in embedded system virtualization, J. Softw. Eng. Appl. 5 (2012) 277–290.
[99] F. Checconi, T. Cucinotta, M. Stein, Real-time issues in live migration of virtual machines, in: Proceedings of the 4th Workshop on Virtualization and High-Performance Cloud Computing (VHPC 2009), Delft, The Netherlands, August 2009.
[100] P. Svard, J., Tordsson, B. Hudzia, E. Elmroth, High performance live migration through dynamic page transfer reordering and compression, in: Proc. of the 3rd IEEE International Conference on Cloud Computing Technology and Science (Cloudcom 2011), IEEE Computer Society, 2011, pp. 542–548.

Marisol García-Valls is associate professor at Universidad Carlos III de Madrid (Spain). Since 2002, she is the head of the Distributed Real-Time Systems Laboratory in the Telematics Engineering Department. Her research interests focus on distributed real-time systems, cyber physical systems, operating systems, service oriented architectures, and reliable middleware technologies. She is Associate Editor of Journal of Systems Architecture. She has been enrolled in a number of European, national, and regional projects as the EU projects ARTIST and ARTIST2, and the European Network of Excellence ARTISTDesign. She is the scientific and technical coordinator of the EU project iLAND (ARTEMIS-1–100026; 2009–2012) and the principal investigator and coordinator of the Spanish national project REM4VSS (TIN-2011–28339; 2012–2014). She has been awarded the 2014 Prize for Outstanding International Research funded by Universidad Carlos III de Madrid, and the Salvador de Madariaga Grant for International Research Projects (PRX12/00252) granted by Spanish Ministry of Education in 2013.

Tommaso Cucinotta is Researcher at Bell Laboratories, Alcatel-Lucent, Dublin (Ireland) since 2012. In 2004, he obtained PhD in Computer Engineering, got at the Scuola Superiore Sant'Anna University, Pisa (Italy), and Master Degree in Computer Engineering in 2000 at the University of Pisa. He has been research associate at the Real-Time Systems Laboratory (RETIS) of the Scuola Superiore Sant'Anna University (SSSA), Pisa (Italy) since October 2011, and assistant professor at the same institution since 2005. He has been involved in PC of different workshops and conferences, as well as reviewer of different relevant journals. He has participated in a number of international an national research projects.

Chenyang Lu is a Professor of Computer Science and Engineering at Washington University in St. Louis. Professor Lu is Editor-in-Chief of ACM Transactions on Sensor Networks and Associate Editor of Real-Time Systems. He has also served as Program Chair of IEEE Real-Time Systems Symposium (RTSS 2012), ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS 2012) and ACM Conference on Embedded Networked Sensor Systems (SenSys 2014). Professor Lu is the author and co-author of over 100 research papers with over 12000 citations and an h-index of 50. He received the Ph.D. degree from University of Virginia in 2001, the M.S. degree from Chinese Academy of Sciences in 1997, and the B.S. degree from University of Science and Technology of China in 1995, all in computer science. His research interests include real-time systems, wireless sensor networks and cyber-physical systems.